# Bolt Beranek and Newman Inc.

Report No. 3753

## Distributed Computation and TENEX-Related Activities

Quarterly Progress Report No. 12, 1 August 1977 to 31 October 1977



January 1978

Prepared for:
Defense Advanced Research Projects Agency

BBN Report No. 3753

# DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Quarterly Progress Report No. 12
1 August 1977 to 31 October 1977

January 1978

Sponsored by:

Defense Advanced Research Projects Agency
ARPA Order No. 2935

Monitored by:

Office of Naval Research
Under Contract No. N00014-75-C-0773
Contract Period 1 November 1974 to 1 May 1978

Principal Investigators: Robert H. Thomas, Richard E. Schantz

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 3753 (14)
2. GOVT ACCESSION NO.: BBN-3753
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): (6) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES.
5. TYPE OF REPORT & PERIOD COVERED: 8/1/77 - 10/31/77
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): (10) Richard E. Schantz, Robert H. Thomas
8. CONTRACT OR GRANT NUMBER(s): (15) N00014-75-C-0773, ARPA Order-2935
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc, 50 Moulton Street, Cambridge, Massachusetts 02138
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: (9) Quarterly progress rept. no. 12, 1 Aug - 31 Oct 77, (11)
12. REPORT DATE: Jan [redacted] 78
13. NUMBER OF PAGES: 45 (12) 48 pp.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distributed computation
National Software Works
distributed operating system
TENEX operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes BBN efforts in the design of the National Software Works system and BBN efforts to integrate TENEX into the National Software Works system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

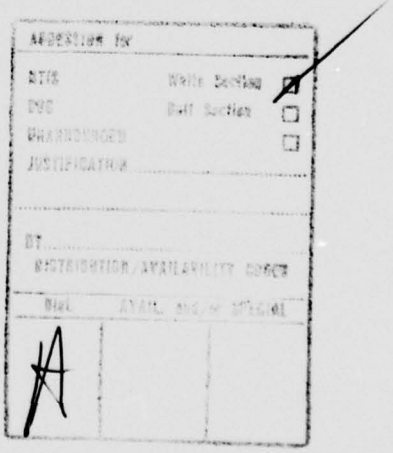
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100

Table of Contents





## 1. Introduction

As in the last quarter, our work this quarter on the National Software Works (NSW) project focused primarily on implementation issues as we continue to work toward the development of an operational NSW system. In general terms, our goal here is to work with the other NSW contractors to develop an NSW system which provides sufficient functionality, has adequate responsiveness, and is resilient to transient failures of system components. Only when NSW has achieved such status can it be regarded and used as an operational system for software production.

In more specific terms our work this quarter has been largely directed toward the system components for which we are responsible: the TENEX (and TOPS-20) implementation of MSG (the NSW interprocess communication facility), the TENEX Foreman (the tool bearing host module responsible for controlling tool execution), and the NSW dispatcher (the component responsible for connecting users to NSW Front Ends as they attempt to access the system).

We completed the conversion of the TENEX MSG implementation to TOPS-20 which we started last quarter. The functionality of MSG was increased to support multi-fork MSG processes; this will be especially useful to system component implementers who find it

convenient to use the fork capabilities of TENEX and TOPS-20 in their implementations. In addition, significant improvements to the reliability of the TENEX (and TOPS-20) MSG implementation were made during this quarter. The details these and other efforts on MSG are presented in Section 2.

Our work on the TENEX Foreman and related tool bearing host software focused on the following areas: improvements to MKCOM, the module that defines and manages Foreman workspace definition files; implementation of a general purpose instrumentation and report generation package which can be used with other NSW components (as well as with the Foreman) to gather component and system performance data; design and implementation of a rudimentary tool-descriptor tool; installation of several TENEX software packages as additional NSW tools; providing assistance to SRI in the installation of NLS as an NSW tool; and the release and maintenance of several new and improved versions of the TENEX Foreman. This and other related work is discussed in detail in Section 3.

In addition to these system implementation efforts we also attended a number of meetings and conferences during this quarter. In August we presented a paper on Network Operating Systems at the Second Brown University Conference on Distributed Processing; The paper, which describes NSW and compares it with other network operating systems, is available as BBN Report No. 3614, titled "Operating Systems for Computer Networks". A

shorter version of it is scheduled to appear in the January issue of the IEEE Computer magazine. We met twice at BBN with personnel from the University of Texas at Austin to discuss NSW performance and plans to develop analytic and simulation models which will be used to study the performance of the NSW system. In September we hosted a two day meeting of NSW contractors involved in system implementation to discuss the reliability plan (MCA Report No. CA-7701-1411) developed for the NSW. This meeting proved to be a valuable one as a number of implementation issues, in addition to system reliability, were addressed and resolved. Regular meetings of this sort are planned for the future. We met with personnel from Rome Air Development Center to discuss the problem of configuration control for the NSW system and also the possibility of developing a control center for operational NSW systems.

Finally, we conducted a series of measurements on the performance of NSW system components. These measurements are similar to the ones performed in April and were made to compare the performance of the most recent NSW system, which includes implementation of the so-called "interim reliability scenarios" and a number of other functional improvements with the predecessor system. The results of these measurements are summarized in Section 4.

## 2. MSG

In the last quarter we started to modify the TENEX implementation of MSG so that it can run under the TOPS-20 operating system. This is one of a number of steps required to integrate the TOPS-20 operating system into NSW (See BBN Report No. 3752 for a discussion of the other steps). We completed this conversion process and now have a single executable module for MSG which can run under either TENEX or TOPS-20. As part of its initialization procedure this module determines and remembers which operating system is being used and then makes run time decisions whenever necessary to execute operating system dependent code.

The TENEX version of MSG uses the JSYS trap mechanism as the means for gaining control when a process executes an MSG primitive. The initial version of TOPS-20 MSG did not use JSYS traps because at the time the mechanism was not available on TOPS-20. Another less flexible mechanism was used. Recently, TOPS-20 has been enhanced to support JSYS traps and the current version of MSG now uses the trap mechanism both on TENEX and TOPS-20.

As part of our performance measurement activities (See BBN Reports No. 3751 and 3752) we developed two processes, M1 and M2, for measuring MSG delays. These processes cooperate to measure

MSG performance as seen by user processes by exchanging messages and alarms and by establishing and breaking MSG direct connections. The specification for the M1/M2 processes (See Appendix A, BBN Report No. 3752) has become a "standard" in that each host which is part of the NSW is expected to implement an M1 and an M2 program to support MSG performance measurements. During this quarter we modified the TENEX programs for M1 and M2 so that these programs can run under both TOPS -20 and TENEX MSG.

Early versions of MSG for TENEX had a limitation that allowed only the top fork in an MSG process to execute MSG primitives. The impact of this was that NSW component implementers choosing to use multiple forks in their implementations had to ensure that only the top fork performed MSG operations. This was a significant restriction because it greatly limited the flexibility with which the multiple fork feature of TENEX could be used. During this quarter this limitation has been eliminated. MSG now supports multiple fork processes in the sense that any fork in an MSG process can execute MSG primitives. This enhancement to MSG is important because it greatly increases the flexibility implementers of NSW components have in structuring their implementations.

The principal modification to MSG needed to remove this limitation was to implement the additional bookkeeping necessary to remember which fork executed a given primitive. For example, when a message arrives for a process and there is an outstanding

receive operation, MSG must deliver the message to the fork in the process that executed the receive primitive. The implementation problem here derives from the facts that in TENEX and TOPS-20 fork handles (names) are relative, and that a fork may have only a small number of active handles for other forks. The fork handles are relative in that the names two forks use to refer to the same third fork are, in general, different and are specific to the referring fork. The number of active handles is small in that it is less than the maximum number of forks allowable in a job. Consequently, the processes MSG is controlling may consist of more forks than it can simultaneously have handles for. To support the execution of primitives by multiple forks in a process MSG was modified to carefully manage the fork handles it uses, acquiring handles for other forks only as needed and releasing them when no longer needed.

A number of problems with the TENEX implementation of MSG direct connections have been corrected this quarter. The TENEX implementation of MSG has evolved as a series of steps, with each step incorporating additional functionality and bringing the implementation closer to the MSG design specification. Direct connections were first implemented and debugged as one of the later steps. The MSG implementations for the other NSW hosts (Multics, the 360/91) were accomplished in a similar manner, with direct connections being added only very recently to these implementations.

As the direct connection feature became available on Multics and the 360/91, problems in the TENEX implementation not uncovered by TENEX-to-TENEX interactions came to our attention. These problems were largely in the areas of closing connections and in rejecting attempts to establish connections. They remained undetected primarily because the TENEX implementation generates only a relatively small number of the many possible sequences of events that can occur as a connection is closed or rejected. As the other hosts began to open and close connections other sequences of events occurred, and in some cases were not properly handled by the TENEX implementation. At present all known problems in the TENEX and TOPS-20 implementations of MSG direct connections have been corrected.

The overall reliability of the TENEX MSG implementation has been significantly improved during this quarter. The improvements are largely the result of experience gained from the use of MSG in the so-called "user" NSW system. From the point of view of system debugging the quasi-operational environment provided by the user NSW system, while less controlled than a typical debugging environment, has two advantages over it.

First, because the user system is used in a realistic manner over long periods of time, much more of the code is exercised, and in more ways, than is generally feasible in a debugging environment. Consequently, many problems which remain undetected during normal debugging activity can be exposed, diagnosed, and

corrected. For example, procedures for handling errors and other abnormal situations are typically not thoroughly exercised in debugging sessions, in part, because it is difficult to anticipate all of the error conditions that are likely to occur.

The second advantage is that certain conditions which are difficult or impossible to produce or to accurately simulate during debugging sessions occur during the "normal" operation of the system. For example, it is impractical to cause large hosts, such as a TENEX which supports many projects in addition to NSW, to crash repeatedly in order to thoroughly debug code (in a remote MSG) designed to handle remote system crashes. However, over several weeks it is likely that one or more of the hosts which support the user NSW system will crash a number of times in a variety of ways, affording system personnel the opportunity to determine whether crash recovery procedures operate properly. We, in fact, discovered that the TENEX Network Control Program (NCP) behaves differently than we expected when a remote host with which it had been communicating crashes. This forced us to modify the code in MSG designed to make it resilient to crashes of remote hosts with which its local processes were communicating.

One of the difficulties in using the user system for debugging is that because it generally runs unattended, when a software problem occurs there is the possibility that by the time it is noticed all useable traces of its cause may have been

obliterated by the continued operation of the system. The MSG implementation addresses this problem by frequently performing internal consistency checks. Upon detecting an inconsistency it executes a procedure, called ERRHLT, whereby it stops, generally before the evidence can be destroyed. When systems personnel later notice that the system has stopped, they can begin to diagnose the problem. The ERRHLT mechanism was very useful in debugging the initial implementation of MSG and has proven invaluable in detecting and correcting problems that occur in the user system.

## 3. The TENEX Foreman

The TENEX tool bearing host software uses a data base for managing the directories used as tool workspaces. This data base contains the current status of the various workspaces. It is used by Foreman processes to allocate and deallocate workspaces as tools are started and stopped, and to maintain on a "rerun list" those tool sessions that could not be completed (See BBN Report No. 3751). The data base is initialized by an interactive program called MKCOM (for MaKe COMmon data base). Among other things, MKCOM allows tool bearing host personnel to define the file directories to be used for tool workspaces. BBN Reports 3451 and 3736 include more complete discussions of MKCOM.

During this quarter we improved MKCOM with the addition of a feature to allow NSW operators to determine and modify the status of the various workspaces. This addition was motivated primarily by a problem that occurs when all workspace directories on the host are used for saved tool sessions. The problem is that no new tools can be started on the host until some of the workspaces are released, either by users who "rerun" the incomplete tool sessions or by system personnel who explicitly "delete" saved tool sessions.

When a tool bearing host is restarted following a crash, NSW system personnel can now use MKCOM to determine the status of the

host's workspaces. If sufficient free workspaces exist, the Foreman can be restarted immediately. If not, then MKCOM can be used to release some or all of the saved workspaces before restarting the Foreman. System personnel would probably choose to release those saved workspaces that contain no files first, followed by the oldest saved workspaces. Since MKCOM observes the same mutual exclusion protocols used by Foreman processes for accessing the workspace data base, the NSW operator has the option of running MKCOM concurrently with the Foreman should he prefer to do so.

Because this procedure requires manual intervention by system personnel, and because it has the undesirable effect of destroying files users may wish to retrieve, it must be regarded as a short term solution to the workspace shortage problem. The two level workspace scheme, described in our last quarterly progress report but not yet implemented, represents what we believe to be the proper long term solution to this problem.

The following typescript illustrates the use of MKCOM to manually manage tool workspaces. User input is underlined:

```
@mkcom.SAV;10

Do you want to modify the state of workspaces?  (Y or N): Y

What is the name of the file to be examined?
(the default is FORCOMFILE.SHR) : forcOMFILE.SHR;1400

Is this file now actively being used
     by a running NSW?  (Y or N): N

4 workspaces out of a total of 10 are currently taken.
Do you want to free some holding saved sessions?  (Y or N): Y
```

```
Workspace NSW-WSD53 is holding a session with ID# 883574080
     (not yet reported to the WM)
     Do you want to free it?  (Y or N): Y

Workspace NSW-WSD55 is holding a session with ID# 883579552
     reported to WM 20-Jan-77 19:11:08
     Do you want to free it?  (Y or N): N

Workspace NSW-WSD57 is holding a session with ID# 883583480
     (not yet reported to the WM)
     Do you want to free it?  (Y or N): N

Workspace NSW-WSD58 is holding a session with ID# 883583672
     reported to WM  1-Aug-77 20:37:37
     Do you want to free it?  (Y or N): Y
```

As part of our efforts in the area of performance monitoring and improvement we have developed a general purpose instrumentation and data analysis package. This package is based on software developed previously to monitor Foreman performance. We started with this Foreman-specific software and removed Foreman-specific, as well as NSW-specific, aspects from it and generalized its interface to permit the functions it supports to be invoked from programs written in higher level languages. At present the package supports BCPL and assembly language interfaces and can easily be augmented to support interfaces to other languages, such as FORTRAN or LISP.

The instrumentation package supports two kinds of measurement functions. It can be used to record the occurrence of events, such as the invocation of a subroutine, and it can be used to perform measurements on intervals, such as the interval between the beginning and end of the execution of a subroutine. The particular events to be recorded are defined by the user. On

the occurrence of each such defined event the particular event along with the time of occurrence is recorded. For interval measurements the user defines the intervals of interest by specifying their beginning and end. At the beginning and end of each such interval the instrumentation package records the real time, CPU time and various other data relevant to performance, such as page fault information.

To instrument a module the measurement package must be loaded with the module. In addition, the module must itself be modified slightly to include calls upon the instrumentation package to invoke the various data collection functions. The instrumentation package uses part of the module address space to store data it gathers. The module can specify the region of its address space to be used for this purpose by calling an instrumentation initialization function. To permanently record the data, the module can call an instrumentation output function that writes the accumulated data onto a file.

The data files produced by the instrumentation package contain raw binary data. Data analysis software has been developed to process this raw data and produce a textual summary of the data. This software takes as input a a series of binary data files produced by the instrumentation package and a text file which specifies the events which are of interest for the summary. For recorded events the user can specify the particular event types to be processed, and for each event type whether he

wants a listing of each occurrence or merely a count of the occurrences or both. For the interval measurements he can specify the particular types of intervals to be processed, and for each interval type whether he wants a listing of every interval measured or an average of the interval data recorded or both. The output for each interval includes the elapsed real and CPU times as well as other data, such as the system load average sampled and the number of page faults occurring during the interval.

We developed this general purpose instrumentation and data analysis software to enable other NSW component implementers to make the same kinds of performance measurements we have been making for the TENEX Foreman. However, we believe that this software is sufficiently general to be used to instrument any TENEX (or TOPS-20) program.

During the quarter the new tools installed included MACRO, IDDT, and DESCRIBE. MACRO is the assembler for the PDP-10. IDDT is a assembly language debugger for the PDP-10 which runs under TENEX and TOPS-20. IDDT was developed at BBN under ARPA support. It is quite similar to, but considerably more sophisticated than, the standard DDT debugger supported by DEC. It differs from DDT in two regards: it runs in a fork superior to the one being debugged and is thus "invisible" to the fork in the sense that it does not interfere with and cannot itself be damaged by that fork's operation; and it can be used to debug multi-fork

programs. With the installation of MACRO and IDDT, NSW now supports the edit/compile/debug cycle for PDP-10 assembly language programmers. This brings to three the languages for which the edit/compile/debug cycle is supported. The other two are BCPL and CMS-2. This is another indication that, in terms of functionality, NSW is evolving toward an operational system for software production.

DESCRIBE is a tool which we designed and implemented specifically for NSW. It is a "tool description tool" which is used to obtain on-line information about other NSW tools. Our initial objective in developing DESCRIBE was somewhat modest. It was to develop a simple program to help us keep track of the TENEX tools we install into NSW. However, it quickly became apparent that such a tool would have general utility in the context of the entire NSW system as a means to provide users with convenient on-line access to tool information.

The following typescript from a session with the DESCRIBE tool illustrates some of its current capabilities. User input is underlined:

```
Type HELP (carriage return) for help.
>HELP

Describe is a tool for describing the
operation of other tools in the NSW environment.
The basic syntax for requesting a description is
"DESCRIBE (Toolname)". Other commands are as follows:
Type ^O to force an interrupt and return to the top level
 "    ^A to backspace one character
 "    rubout(DEL) to abort the command string being typed in
 "    ? (carriage return) to see a list of options at any point
>?
```

```
DESCRIBE
QUIT
HELP
>DESCRIBE ?
BCPL
BDDT
FTP
HOSTAT
IDDT
JIGSAW
LINKER
MACRO
MRUNOFF
NETSTAT
SPELL
>DESCRIBE MACRO
```

MACRO is the PDP-10 assembly language assembler. Its use in the NSW environment is unrestricted. However, a file spec may have a maximum of two NSW filename fields. The first field can have no more than 6 characters, and the second no more than 3. Upon starting the tool the user receives a prompt character "*". He then types the name for the REL file to be created, followed by a left arrow or underscore (depending on the terminal character set), followed by the source file name. If the second name field is omitted, the tool will use REL and source files with the default fields .REL and .MAC respectively.

Example: *TEST_TEST

would assemble TEST.MAC placing the output in the file TEST.REL. Extensive documentation is provided in both the Tenex Users Guide and in the DEC-10 Users Handbook. Tool termination is accomplished by typing ^Z. Please direct all inquiries to COOK@BBN.

```
>DESCRIBE IDDT
```

IDDT is an interactive assembly language level debugger for Tenex. The user may yank in a save file by typing ;Y. Similarly, making a copy of the core image (SSAV'ing it) is done by typing ;U. Documentation is available in both the Tenex Users Guide and in the DEC-10 Users Handbook. Termination of the tool is accomplished by typing ;H. Please direct all inquiries to COOK@BBN.

```
>QUIT
```

At present DESCRIBE, while very useful, is best regarded as a prototype for a far more general tool description tool. To

transform DESCRIBE into a general NSW tool description tool its user interface, as well as its internal structure, would be enhanced to support selective access to a variety of different types of tool information. DESCRIBE would obtain the actual information required to satisfy users' requests from a tool description data base. The tool specific data would be submitted by tool implementers and installers as part of the tool installation procedure and would be entered into the DESCRIBE data base by NSW personnel.

Currently further development of DESCRIBE is not a specific task item in our contract. However, we believe that development of DESCRIBE along the lines suggested above would be a significant contribution to the evolution of NSW into an operational system, and we recommend it highly.

Several new versions of the TENEX Foreman were released and integrated into the NSW system. In addition to bug fixes, these releases included a number of changes and improvements, some of which are described in the following paragraphs.

The current version of the Foreman now collects samples of its own page faulting statistics during tool execution, as part of its performance monitoring activities. To collect and process this data, in addition to the other performance data described in previous quarterly progress reports, the instrumentation and analysis programs were modified, and event calls were inserted into a new Foreman at appropriate sampling locations.

As discussed both in this and previous reports, the Foreman maintains tool workspaces in a crash proof manner in order to save workspaces for tool sessions interrupted by crashes. As part of its crash recovery procedure the Foreman reports any new workspaces that must be saved to the Works Manager. Previously, the Foreman would delete workspaces it discovered corresponding to interrupted tool sessions that do not contain files. The reasoning was that there was nothing of interest to be saved for the user. At present the Works Manager is not prepared to distinguish between workspaces which are saved because they hold files and those which are deleted because they do not. The short term solution to this problem was to modify the Foreman to save all workspaces, even if they are empty, until the user attempts to rerun them. The correct long term solution is to modify the Works Manager to be able to process a message which distinguishes between empty and file-containing workspaces so that the Foreman need not waste the storage and CPU time required to save them, and the user the effort to rerun them.

During tool execution the Foreman maintains, as part of the local name dictionary (LND) a "local file list" which contains the names of non-NSW files resident on the Foreman host that the tool may directly reference; an example of such a file is the standard dictionary file used by SPELL. This list is used by the Foreman in determining how to handle file open operations initiated by the tool. The list is searched for the file referenced. If both that search and a search of the rest of the

LND fails, the file reference is passed to the Works Manager in order to complete the open operation.

For early versions of the Foreman the local file list was static in the sense that no additions could be made to it as the tool executed. This caused a problem for some tools which make use of TENEX temporary (";T" and ";S") files. The problem occurred when, if after creating and closing such a file, the tool would attempt to re-open it without re-specifying that the file was temporary; that is, without including the ";T" or ";S" in the file name. If the tool were running directly under TENEX, the open would succeed. However, since the second reference to the file omits indication the file is temporary, the Foreman would use its normal file lookup procedure. It would search the local file list and then the rest of the LND and, since it would not be found, would forward the open request to the Works Manager, which would not find it either. As a result the open would fail. The current version of the Foreman corrects this problem by dynamically adding temporary files to the local file list when they are created. This ensures that subsequent tool references to them will succeed.

Finally, we have interacted extensively with SRI regarding the installation of NLS as an NSW tool. The current approach for installing NLS is through the use of a specialized Front End that is capable of driving a message oriented NLS back end which serves as the "tool". Communication between these modules is to

be supported by a pair of binary connections (see BBN Report 3752). File system support for the back end processor is to be handled through encapsulation in the normal fashion.

This quarter we have worked with SRI to integrate their Front End with the TENEX Foreman in running the standard NSW tool set, and additionally in debugging the binary communication support. We have also had numerous discussions regarding appropriate strategies to make NSW NLS more compatible with encapsulation. Their implementation effort has uncovered a number of areas which require further development to obtain a more accurate encapsulating environment. These considerations will be part of the future implementation efforts for the encapsulation module.

# 4. NSW Performance Measurements

## 4.1 CPU Measurements for TENEX Components

This quarter we conducted a second series of experiments designed to measure the performance of NSW components in carrying out standard NSW operations. These experiments were similar to the first series of experiments documented in BBN Report 3751. The intent was to see how the performance characteristics for the current components have changed since the introduction of the protocol changes relating to the reliability constructs and through general component improvements. We have added measurement data for other parts of the system configuration which were not present in the previous report. This includes data on a new component (the WM data base Checkpointer), real elapsed time data for some of the trials, and a sampling of the system load average during some of the trials. In addition, we have added a series of trials in some areas in which we turn off any event recording facilities which may be present in the various components being measured to see if these have a significant impact on performance.

This experiment is intended mainly to obtain the CPU time requirements for the components in handling the various scenarios. The measurements were made using the event logging capability of MSG which, when enabled, logs the occurrence of

various events. We have additionally augmented these measurements with data (real time and load average) obtained from the Foreman instrumentation regarding these same test sessions.

The general experimental procedure was the same as for the first series of experiments. An NSW configuration was initialized with MSG event logging enabled and the standard scenarios for user sessions were used to exercise the system. With a knowledge of the user scenarios and the underlying inter-component protocols, the resulting MSG event log was analyzed to determine component CPU times for the various NSW operations. The event log also indicates which Foreman process was involved with which scenario instance, and from that we can match the tool session data with the experimental trial.

In general, these measurements reflect internal component CPU requirements exclusive of MSG CPU requirements for inter-component communication support. For "servers" (e.g. Works Manager) the CPU quantity is generally measured from receipt of a request to completion of that request. For components that generate requests, CPU time is generally measured from initiation of a request until the request is satisfied. The exact intervals measured, along with a brief semantic meaning associated with each component interaction are described immediately preceding the data for each scenario. The averages for the experiments run in April are included for comparison purposes.

All experiments were performed with only a single active NSW user. The measurements were made on the BBNB TENEX for an NSW system usually used for debugging purposes. The configuration was run in a 12% pie-slice, but it did not necessarily have exclusive access even to this share. This was a single host configuration. There were no inter-host messages.

The following "certification" data serves to identify the components measured in these experiments.

Works Manager: CERTIFICATION:
r.faneuf,sri-ka,<WMSRC>WM.SAV;165,27-May-77 12:36:08

TENEX File Package: CERTIFICATION:
R.FANEUF,SRI-KA,<FANEUF>FLPKG.SAV;61, 4-May-77 08:20:17

TENEX Front End: CERTIFICATION:
bearisto,sri-ka,<BEARISTO>FE.SAV;433, 27-May-77 11:06:29

TENEX Foreman: CERTIFICATION:
r.schantz,bbn,<BBN-NSWTST>FOREMAN.SAV;1420, 14-Jul-77 16:59:46

WM Database Checkpointer: CERTIFICATION:
r.faneuf,sri-ka,<WMSRC>CHKPTR.SAV;5, 26-May-77 21:44:09

Measurements were made for the RUNTOOL, TOOLHALT, OPEN and DELIVER tool operations, and for the COPY and DELETE NSW EXEC file operations. Common ID numbers for the RUNTOOL, TOOLHALT, OPEN and DELIVER scenario trials correspond to the same tool session. ID numbers for the DELETE and COPY trials are unrelated to any other trials.

In summary, the experiments show that the average total component CPU requirements for the various scenarios has improved slightly (DELIVER is the notable exception). This is encouraging

from two standpoints. First, any improvement, even a slight one, indicates that component implementors are now considering system performance as an important part of their implementation. Second, the improvements are actually a bit larger than a strict comparison of this experiment with those of April would show. This is because of the added functionality associated with the reliability plan which is incorporated into the current components. However, the CPU requirements (and hence the real time performance) are still intolerably high, and much work remains to alleviate this obvious bottleneck.

The experimental data follows.

RUNTOOL SCENARIO:

| | | |
|---|---|---|
| 1. SendGeneric | FE -> WM | (requesting tool invocation) |
| 2. SendGeneric | WM -> FM | (selecting FM to run tool) |
| 3. SendSpecific | FM -> WM | (reporting the tool workspace) |
| 4. SendSpecific | FM -> FE | (initiating connection protocol) |
| 5. OpenConn | FM -> FE | (MSG OpenConnection) |
| 6. OpenConn | FE -> FM | (MSG OpenConnection) |
| 7. SendSpecific | WM -> FE | (reporting selected FM) |

CPU quantities measured (in milliseconds):

FE: Time from execution of SendGeneric (1) to execution of ReceiveSpecific following (7).

WMT: Time from receipt of generic message (1) to execution of ReceiveGeneric following (7).

WM1: Time from receipt of generic message (1) to execution of SendGeneric (2).

WM2: Time from receipt of specific message (3) to execution of SendSpecific (7).

FM: Time from receipt of generic message (2) to completion of OpenConn (5).

TOTAL: FE+WMT+FM

(The following set of trials were taken with both the WM and FM logging facilities on, and under a fairly high TENEX system load)

| ID | FE | WMT | WM1 | WM2 | FM | TOTAL |
|---|---|---|---|---|---|---|
| 1 | 353 | 3700 | 1922 | 1725 | 1556 | 5609 |
| 2 | 347 | 3692 | 1881 | 1753 | 1120 | 5159 |
| 3 | 323 | 3705 | 1905 | 1750 | 1184 | 5212 |
| 4 | 290 | 3664 | 1916 | 1720 | 1040 | 4994 |
| 5 | 1243 | 3892 | 1999 | 1803 | 1271 | 6406 |
| 6 | 723 | 3810 | 1919 | 1812 | 1168 | 5709 |
| 7 | 941 | 3760 | 1960 | 1716 | 1208 | 5909 |
| 8 | 653 | 4203 | 2082 | 2049 | 1204 | 6060 |
| 9 | 353 | 3652 | 1877 | 1710 | 1140 | 5145 |
| 10 | 399 | 3776 | 1938 | 1784 | 1274 | 5449 |
| 11 | 529 | 3798 | 1968 | 1762 | 1371 | 5698 |
| 12 | 517 | 3734 | 1934 | 1768 | 1409 | 5660 |
| 13 | 352 | 3745 | 1931 | 1741 | 1331 | 5428 |
| 14 | 325 | 3764 | 1954 | 1738 | 1382 | 5471 |
| 15 | 348 | 3682 | 1932 | 1698 | 1563 | 5593 |
| AV | 513 | 3771 | 1941 | 1768 | 1281 | 5566 |

Average for similar measurements taken in April 1977:

| | | | | | |
|---|---|---|---|---|---|
| 522 | 4784 | 2146 | 2599 | 1309 | 6615 |

The elapsed time for the above trials is measured in seconds and is based on the interval from the Foreman receiving the Generic message (2) until the tool is ready for use (6). The user will perceive a somewhat lengthier delay because of FE and WM processing prior to (2) and subsequent to (6).

| ID | real time | system ld av | group ld av |
|---|---|---|---|
| 1 | 36.79 secs | 3.43 | 2.01 |
| 2 | 53.32 | 4.17 | 3.28 |
| 3 | 26.80 | 3.80 | 3.74 |
| 4 | 14.85 | 3.13 | 3.20 |
| 5 | 291.75 | 11.70 | 7.71 |
| 6 | 116.82 | 12.09 | 8.09 |
| 7 | 115.14 | 14.70 | 11.98 |
| 8 | 134.51 | 7.38 | 6.12 |
| 9 | 36.29 | 4.82 | 4.60 |
| 10 | 25.15 | 3.13 | 2.79 |
| 11 | 62.20 | 5.44 | 5.27 |
| 12 | 48.67 | 4.54 | 5.00 |
| 13 | 37.77 | 5.62 | 6.37 |
| 14 | 27.03 | 4.47 | 4.93 |
| 15 | 19.27 | 3.53 | 3.01 |

(The following RUNTOOL trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | FE | WMT | WM1 | WM2 | FM | TOTAL |
|---|---|---|---|---|---|---|
| 1 | 296 | 3494 | 1819 | 1609 | 1159 | 4949 |
| 2 | 302 | 3429 | 1772 | 1572 | 868 | 4599 |
| 3 | 258 | 3466 | 1842 | 1596 | 1181 | 4905 |
| 4 | 252 | 3440 | 1793 | 1608 | 1179 | 4871 |
| 5 | 233 | 3428 | 1832 | 1571 | 1226 | 4887 |
| 6 | 234 | 3442 | 1815 | 1618 | 1269 | 4945 |
| 7 | 308 | 3418 | 1792 | 1563 | 1215 | 4941 |
| 8 | 358 | 3605 | 1847 | 1696 | 1072 | 5035 |
| 9 | 331 | 3564 | 1846 | 1656 | 1113 | 5008 |
| AV | 286 | 3476 | 1818 | 1610 | 1142 | 4904 |

TOOLHALT SCENARIO:

| | | |
|---|---|---|
| 1. SendSpecific | FM -> FE | (indicate tool halt; pass tool charges) |
| 2. SendGeneric | FE -> WM | (indicate tool halt; pass tool charges) |
| 3. CloseConn | FE -> FM | (MSG close connection) |
| 4. CloseConn | FM -> FE | (MSG close connection) |
| 5. SendSpecific | WM -> FE | (display tool charges to user) |
| 6. SendSpecific | FE -> WM | (acknowledge tool charges display) |
| 7. SendSpecific | WM -> FE | (acknowledge tool halt message) |
| 8. SendSpecific | FE -> FM | (acknowledge tool halt) |
| 9. SendSpecific | CP -> FM | (guarantee of saved filenames) |

Note: CP is the Checkpointer, a WM subcomponent which periodically saves a copy of the WM data base, and then alerts any other components to this fact. Because of its global nature, its measurement data is reported in a separate section following the TOOLHALT data. For all measurements below, the tool termination is initiated by the tool itself (e.g., TECO ";h", MACRO "^Z").

CPU quantities measured (ms):

FE: Time from receipt of specific message (1) to execution of ReceiveSpecific following (7).

WMT: Time from receipt of generic message (2) to execution of ReceiveGeneric after (7).

WM1: Time from receipt of generic message (2) to execution of SendSpecific (5).

WM2: Time from receipt of specific message (6) to execution of SendSpecific (7).

FMT: Time from execution of SendSpecific (1) to StopMe following (9).

FM1: Time from execution of SendSpecific (1) to receipt of specific message (8).

FM2: Time from receipt of specific message (8) to execution of StopMe following (9).

TOTAL: FE+WMT+FM1+FM2.

(The following set of trials were taken with both the Foreman and Works Manager logging facilities on, and under a fairly high TENEX system load)

| ID | FE | WMT | WM1 | WM2 | FMT | FM1 | FM2 | TOTAL |
|---|---|---|---|---|---|---|---|---|
| 1 | 423 | 4123 | 2321 | 1756 | 1472 | 23 | 1449 | 6018 |
| 2 | 458 | 4135 | 2286 | 1800 | 818 | 40 | 678 | 5311 |
| 3 | 613 | 4273 | 2410 | 1824 | 804 | 79 | 725 | 5690 |
| 4 | 883 | 4438 | 2404 | 1961 | 779 | 92 | 687 | 6100 |
| 5 | 690 | 4452 | 2467 | 1904 | 782 | 112 | 670 | 5914 |
| 6 | 564 | 4379 | 2464 | 1838 | 941 | 126 | 815 | 5884 |
| 7 | 707 | 4398 | 2367 | 1970 | 858 | 99 | 759 | 5963 |
| 8 | 546 | 4173 | 2313 | 1806 | 733 | 82 | 651 | 5452 |
| 9 | 444 | 4173 | 2288 | 1833 | 805 | 46 | 759 | 5422 |
| 10 | 544 | 4257 | 2323 | 1914 | 788 | 58 | 728 | 5587 |
| 11 | 610 | 4204 | 2318 | 1831 | 811 | 150 | 661 | 5626 |
| 12 | 515 | 4197 | 2317 | 1826 | 762 | 99 | 663 | 5474 |
| 13 | 662 | 4359 | 2392 | 1901 | 648 | 84 | 564 | 5669 |
| 14 | 464 | 4296 | 2341 | 1906 | 752 | 49 | 703 | 5512 |
| 15 | 825 | 4437 | 2443 | 1954 | 718 | 78 | 640 | 5980 |
| AV | 596 | 4286 | 2364 | 1868 | 831 | 81 | 743 | 5707 |

Average for April 1977 data (protocols for TOOLHALT have changed sufficiently to make the averages not comparable; the old average is reported for completeness only):

| | | | | |
|---|---|---|---|---|
| 549 | 5390 | | 492 | 6922 |

The elapsed time for the above trials is measured in seconds and is based on the interval from the Foreman SendSpecific (1) initiating the termination until the receipt of the FE reply (8) noting the table updates. The wait period for the guarantee message (9) is not included.

| ID | real time | system ld av | group ld av |
|---|---|---|---|
| 1 | 49.03 secs | 4.90 | 3.46 |
| 2 | 55.48 | 5.88 | 4.49 |
| 3 | 82.63 | 4.17 | 3.88 |
| 4 | 264.25 | 4.68 | 3.64 |
| 5 | 270.68 | 18.37 | 17.10 |
| 6 | 297.43 | 13.80 | 9.90 |
| 7 | 138.05 | 6.30 | 7.33 |
| 8 | 76.79 | 7.31 | 6.77 |
| 9 | 47.56 | 3.29 | 3.20 |
| 10 | 86.33 | 6.79 | 6.63 |
| 11 | 85.77 | 6.47 | 6.57 |
| 12 | 94.64 | 10.62 | 9.31 |
| 13 | 99.24 | 6.91 | 8.08 |
| 14 | 57.19 | 2.93 | 2.88 |
| 15 | 138.13 | 3.35 | 2.66 |

(The following TOOLHALT trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | FE | WMT | WM1 | WM2 | FMT | FM1 | FM2 | TOTAL |
|---|---|---|---|---|---|---|---|---|
| 1 | 319 | 4006 | 2200 | 1750 | 258 | 46 | 212 | 4583 |
| 2 | 354 | 3924 | 2184 | 1732 | 257 | 35 | 222 | 4535 |
| 3 | 342 | 3986 | 2277 | 1692 | 192 | 29 | 163 | 4540 |
| 4 | 258 | 3976 | 2233 | 1727 | 277 | 30 | 247 | 4511 |
| 5 | 348 | 4077 | 2257 | 1777 | 262 | 27 | 235 | 4687 |
| 6 | 388 | 3977 | 2228 | 1747 | 223 | 18 | 205 | 4588 |
| 7 | 507 | 4053 | 2206 | 1763 | 289 | 43 | 246 | 4849 |
| 8 | 524 | 4059 | 2218 | 1771 | 243 | 39 | 204 | 4826 |
| 9 | 427 | 4176 | 2318 | 1808 | 240 | 28 | 212 | 4843 |
| AV | 385 | 4026 | 2236 | 1752 | 249 | 33 | 216 | 4660 |

CHECKPOINTER performance

The CHKPTR is a companion module to the WM component, and is responsible for periodically checkpointing (i.e. saving a consistent copy) the WM data base. Operationally, the CHKPTR is a cyclic program which awakens (currently every 10 minutes of real time), creates a new file image of the data base, and sends a guarantee message to those FM who have terminated since the last checkpoint. The CHKPTR then dismisses for the specified cycle time. In measuring the operational behavior of the CHKPTR module we have established two data points:

"a") the approximate CPU time per guarantee message to an FM

"b") the approximate CPU time to checkpoint the data base

Interval "a" is computed based on the CPU difference between sending consecutive guarantee messages (within a single cycle) when more than one such guarantee is pending at the time of checkpointing. Interval "b" is computed based on the CPU difference between sending the first guarantee message of a cycle and the sending of the last guarantee message of the previous cycle, less the average CPU time for sending one guarantee message (interval "a").
Real elapsed time is computed only for interval "a" (guarantee message time) since the data points for interval "b" include the cyclic wait period.

| | "a": Guarantee | | "b":Checkpointing |
|---|---|---|---|
| ID | CPU (mlsecs) | Real-time (seconds) | CPU (ms) |
| 1 | 469 | 13.29 | 6185 |
| 2 | 533 | 77.26 | 5741 |
| 3 | 514 | 46.69 | 6005 |
| 4 | 519 | 25.80 | 6123 |
| 5 | | | 5720 |
| 6 | | | 5755 |
| 7 | | | 5668 |
| 8 | | | 5398 |
| 9 | 419 | 7.55 | 5361 |
| 10 | 456 | 10.54 | 5460 |
| 11 | 419 | 14.86 | 5123 |
| 12 | | | 5074 |
| 13 | 473 | 30.33 | 5759 |
| 14 | 453 | 17.10 | |
| 15 | 476 | 20.68 | |
| 16 | 456 | 18.04 | |
| AV | 472 ms | | 5644 ms |

(Note: the ID numbers are merely for line identification; no explicit relationship is intended between the "a" trial and

the "b" trial on a given line. Spaces between trial lines indicate separate instances of NSW used to accumulate the data. Load averages varied between 4 and 18, while group load averages varied between 4 and 17.)

OPEN SCENARIO:

| | |
|---|---|
| 1. SendGeneric | FM -> WM (request the NSW file) |
| 2. SendGeneric | WM -> FLPKG (invoke the copy operation) |
| 3. SendSpecific | FLPKG -> WM (reply with file copy name) |
| 4. SendSpecific | WM -> FM (acknowledge with file copy name) |

Note: OPEN is a local copy from NSW file storage space into tool workspace; this is the case because the NSW configuration being measured is a single host system. In all cases the file opened was very small (<50 bytes) and its name was unambiguously specified.

CPU quantities measured (ms):

WMT: Time from receipt of generic message (1) to execution of ReceiveGeneric following (4).

WM1: Time from receipt of generic message (1) to execution of SendGeneric (2).

WM2: Time from receipt of specific message (3) to execution of SendGeneric (2).

FLPKG: Time from receipt of generic message (2) to execution of ReceiveGeneric arter (3).

TOTAL: WMT+FLPKG

(The following set of trials were taken with both the FM and WM logging facilities on, and under a fairly high TENEX system load)

| ID | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|
| 6 | 4295 | 3258 | 943 | 3015 | 7310 |
| 7 | 4269 | 3197 | 995 | 2717 | 6986 |
| 8 | 4372 | 3368 | 944 | 2925 | 7297 |
| 9 | 4224 | 3225 | 949 | 2951 | 7175 |
| 10 | 4549 | 3520 | 953 | 2793 | 7342 |
| 11 | 4451 | 3393 | 979 | 3142 | 7593 |
| 12 | 4426 | 3344 | 1003 | 3338 | 7764 |
| 13 | 4348 | 3301 | 980 | 3073 | 7421 |
| 14 | 4285 | 3292 | 952 | 2924 | 7209 |
| 15 | 4113 | 3153 | 928 | 2926 | 7039 |
| AV | 4333 | 3305 | 962 | 2980 | 7314 |

Average for similar measurements taken in April 1977:

| | | | | |
|---|---|---|---|---|
| 5261 | 4000 | 1602 | 1928 | 7549 |

The elapsed time for the above trials is measured in seconds and is based on the interval from the Foreman initiating the request (1) until receiving notification of its completion (4).

| ID | real time | system ld av | group ld av |
|---|---|---|---|
| 6 | 225.65 secs | 15.23 | 8.41 |
| 7 | 90.50 | 11.62 | 12.75 |
| 8 | 220.12 | 12.26 | 11.35 |
| 9 | 81.69 | 3.27 | 1.75 |
| 10 | 132.19 | 9.13 | 6.67 |
| 11 | 221.29 | 10.35 | 5.60 |
| 12 | 235.00 | 6.80 | 7.07 |
| 13 | 195.54 | 6.19 | 8.01 |
| 14 | 47.35 | 4.37 | 4.67 |
| 15 | 34.82 | 3.91 | 3.62 |

(The following OPEN trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|
| 4 | 4004 | 3132 | 829 | 2616 | 6620 |
| 5 | 4172 | 3304 | 828 | 2622 | 6794 |
| 6 | 4078 | 3170 | 887 | 2534 | 6612 |
| 7 | 4194 | 3227 | 903 | 2582 | 6776 |
| 8 | 4366 | 3398 | 905 | 2879 | 7245 |
| 9 | 4237 | 3324 | 889 | 2758 | 6995 |
| AV | 4175 | 3259 | 873 | 2665 | 6840 |

DELIVER SCENARIO:

| | | |
|---|---|---|
| 1. | SendGeneric | FM -> WM (request the delivery) |
| 2. | SendGeneric | WM -> FLPKG (invoke the copy operation) |
| 3. | SendSpecific | FLPKG -> WM (reply with file copy name) |
| 4. | SendSpecific | WM -> FM (acknowledge completion) |

Note: DELIVER, like OPEN, involves a local file copy. For DELIVER, the copy is from tool workspace into NSW file space. As in the OPEN measurement, the file in question was very short and the file name always unambiguously specified. In addition, all of the DELIVER operations created new files rather than replaced existing ones.

CPU quantities measured (ms):

WMT: Time from receipt of generic message (1) to execution of ReceiveGeneric following (4).

WM1: Time from receipt of generic message (1) to execution of SendGeneric (2).

WM2: Time from receipt of specific message (3) to execution of SendSpecific (4).

FLPKG: Time from receipt of generic message (2) to execution of ReceiveGeneric after (3).

TOTAL: WMT+FLPKG

(The following set of trials were taken with both the Foreman and Works Manager logging facilities on, and under a fairly high TENEX system load)

| ID | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|
| 12 | 4251 | 3370 | 847 | 3986 | 8239 |
| 13 | 4187 | 3365 | 767 | 3979 | 8266 |
| 14 | 4182 | 3360 | 800 | 3901 | 8082 |
| 15 | 4180 | 3327 | 802 | 3924 | 8082 |
| AV | 4200 | 3355 | 804 | 3947 | 8147 |

Average for similar measurements taken in April 1977:

| | | | | |
|---|---|---|---|---|
| 4265 | 2792 | 1452 | 2783 | 7048 |

The elapsed time for the above trials is measured in seconds and is based on the interval from the Foreman initiating the request (1) until receiving notification of its completion (4).

| ID | real time | system ld av | group ld av |
|---|---|---|---|
| 12 | 123.89 secs | 11.96 | 10.26 |
| 13 | 90.74 | 8.82 | 10.21 |
| 14 | 38.87 | 2.67 | 2.52 |
| 15 | 53.02 | 2.51 | 2.12 |

(The following DELIVER trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|
| 7 | 4045 | 3280 | 683 | 3753 | 7798 |
| 8 | 4012 | 3264 | 680 | 3900 | 7912 |
| 9 | 4009 | 3279 | 695 | 3594 | 7603 |
| AV | 4022 | 3278 | 689 | 3749 | 7771 |

DELETE SCENARIO:

| | | |
|---|---|---|
| 1. | SendGeneric | FE -> WM (request the deletion) |
| 2. | SendSpecific | WM -> FE (ask for confirmation) |
| 3. | SendSpecific | FE -> WM (deletion confirmed) |
| 4. | SendGeneric | WM -> FLPKG (invoke file deletion) |
| 5. | SendSpecific | FLPKG -> WM (acknowledge copy deletion) |
| 6. | SendSpecific | WM -> FE (acknowledge deletion) |

Note: The file name was unambiguously specified in all cases.

CPU quantities measured (ms):

FE: Time from execution of SendGeneric (1) to execution of ReceiveSpecific after (6).

WMT: Time from receipt of generic message (1) to execution of ReceiveGeneric after (6).

WM1: Time from receipt of generic message (1) to execution of SendSpecific (2).

WM2: Time from receipt of specific message (3) to execution of SendGeneric (4).

WM3: Time from receipt of specific message (5) to execution of SendSpecific (6).

FLPKG: Time from receipt of generic message (4) to execution of ReceiveGeneric after (5).

TOTAL: FE+WMT+FLPKG

(The following set of trials were taken with both the Foreman and Works Manager logging facilities on, and under a fairly high TENEX system load)

| ID | FE | WMT | WM1 | WM2 | WM3 | FLPKG | TOTAL |
|---|---|---|---|---|---|---|---|
| 1 | 436 | 3487 | 2053 | 586 | 744 | 922 | 4845 |
| 2 | 340 | 3461 | 2015 | 620 | 753 | 868 | 4669 |
| 3 | 470 | 3414 | 2000 | 614 | 742 | 843 | 4727 |
| AV | 415 | 3454 | 2023 | 607 | 746 | 878 | 4747 |

Average for similar measurements taken in April 1977:

| | | | | | | |
|---|---|---|---|---|---|---|
| 373 | 4064 | 1917 | 537 | 1586 | 780 | 5216 |

(The following DELETE trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | FE | WMT | WM1 | WM2 | WM3 | FLPKG | TOTAL |
|---|---|---|---|---|---|---|---|
| 1 | 212 | 2904 | 1903 | 507 | 483 | 872 | 3988 |
| 2 | 277 | 3019 | 1922 | 506 | 584 | 826 | 4122 |
| 3 | 270 | 3125 | 1917 | 505 | 662 | 859 | 4254 |
| AV | 253 | 3016 | 1914 | 506 | 576 | 852 | 4121 |

COPY SCENARIO:

| | | |
|---|---|---|
| 1. | SendGeneric | FE -> WM (request the operation) |
| 2. | SendGeneric | WM -> FLPKG (invoke actual file copy) |
| 3. | SendSpecific | FLPKG -> WM (report file copy name) |
| 4. | SendSpecific | WM -> FE (acknowledge completion) |

Note: The file names were unambiguously specified in all cases.

CPU quantities measured (ms):

FE: Time from execution of SendGeneric (1) to execution of ReceiveSpecific after (4).

WMT: Time from receipt of generic message (1) to execution of ReceiveGeneric after (4).

WM1: Time from receipt of generic message (1) to execution of SendGeneric (2).

WM2: Time from receipt of specific message (3) to execution of SendSpecific (4).

FLPKG: Time from receipt of generic message (2) to execution of ReceiveGeneric after (3).

TOTAL: FE+WMT+FLPKG

(The following set of trials were taken with both the Foreman and Works Manager logging facilities on, and under a fairly high TENEX system load)

| ID | FE | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|---|
| 1 | 490 | 4885 | 3827 | 998 | 3608 | 8983 |
| 2 | 251 | 4711 | 3656 | 1019 | 3557 | 8519 |
| 3 | 204 | 4667 | 3640 | 996 | 3665 | 8536 |
| 4 | 453 | 4885 | 3656 | 1025 | 3668 | 9006 |
| 5 | 361 | 4650 | 3544 | 1034 | 3742 | 8753 |
| AV | 351 | 4760 | 3665 | 1014 | 3648 | 8759 |

No previous measurement data.

(The following COPY trials were taken with both the Foreman and Works Manager logging turned off and under a fairly light Tenex system load)

| ID | FE | WMT | WM1 | WM2 | FLPKG | TOTAL |
|---|---|---|---|---|---|---|
| 1 | 127 | 4378 | 4355 | 928 | 3272 | 7777 |
| 2 | 100 | 4356 | 4352 | 841 | 3420 | 7876 |
| 3 | 156 | 4450 | 4446 | 917 | 3440 | 8046 |
| AV | 128 | 4395 | 4384 | 895 | 3377 | 7900 |

## 4.2 NSW User System Performance Statistics

We have collected and analyzed TENEX Foreman component activity logs generated on the ISIC "user" NSW system over the period from June 1,1977 through August 12,1977. The collected data deals with tool activity over this period, as well as NSW performance data concerning the primary tool related NSW operations: beginning a tool, terminating a tool, retrieving an NSW file, and replacing an NSW file. Because of various peculiarities in the data obtained, we have subjected some of it to somewhat more rigorous analysis than normally might be expected for data of this nature, in an attempt to determine the characteristics of the load which the NSW imposes on its host computer. We are also trying to uncover any operational peculiarities which might be areas for possible optimization or warrant further investigation. In addition, the analysis of the collected data has been part of an iterative procedure in which we are evaluating our data and statistical methods and will often respecify the data to be collected based on the previous analysis. This iteration is still continuing, mainly owing to persisting inadequate understanding and explanation of overall system performance measures.

The nature of the "user" NSW is such that the data collected regarding tool usage patterns may not always reflect accurately a typical situation. However, we think it is useful to report some of our findings now, if only to illustrate the potential of the

mechanisms already in place, and to give a realistic picture of what can be expected when using the system today.

A total of 213 tool sessions were successfully initiated in the above period according to the log files we have been able to retrieve. A successful initiation consists of establishing the tool instance and providing it with a direct connection to the FE. Six different tools were used-- TECO (133 uses), XED (40), <PRIM>EMLOAD (11), <PRIM>UYK20 (11), and <PRIM>MACRO20 (10). The average tool session lasted for 13.6 minutes. In these tool sessions, 193 "get NSW file" requests (i.e. send WM-GET to WM and receive reply) were processed, approximately one file per tool session. There were 123 requests to "put a file into NSW" (i.e. send WM-DELIVER and receive a reply), indicating many sessions saved no files.

Of the 213 successfully initiated tool sessions, about two thirds (143) were completed under normal circumstances:

105 (the predominant mode) were terminated by a tool command
7 were terminated via the FE command for tool termination
31 were aborted via the FE "abort" command.

A breakdown of the 70 abnormal terminations is as follows:

17 Timeout of a WM operation
10 Timeout of a FE operation (including broken connection)
11 Protocol violation or FM program bug
29 SAVE-LND due to broken user connection to his FE
3 Unclassified by the analysis program.

It remains unclear whether this usage pattern represents a given user community, is the result of exercising the facility, or reflects user behavior induced by erratic and oftentimes poor system performance. The eleven cases in which a protocol error or internal program failure was detected have been very valuable, since reports of these tool sessions have permitted the rapid off-line analysis of such unexpected failures, as well as an accurate designation of the parties responsible for their correction.

In addition to measuring the basic tool workload, we have measured both the Foreman process CPU time and the elapsed real time (as seen from the TENEX Foreman component) in carrying out the four basic tool related operations: beginning, file retrieval, file delivery, and termination. As might be expected when operating on a "time-shared" processing facility, the real time to complete a given operation varied quite dramatically over the recorded trials. Somewhat surprisingly, the system load average did not provide a particularly good measure in predicting the real time delay. (Underlying our expectations that it might provide a good measure were assumptions that for the most part there was a single tool interacting with NSW at any given time [note: the 213 tool sessions were spread over 52 working days]; that the "pie-slice" devoted to the NSW during the tool sessions remained fairly constant; and that the operating system configuration during the trial period did not undergo drastic reconfiguration. The analysis programs are not yet able to

capture and verify these details. However, we are now exploring ways to factor these and other measures into our performance data.) By and large, a very high system load usually produces a long delay, and a very low system load average usually produces much shorter delays. Nonetheless, the analysis of the data seems to emphasize that there are important factors other than system load average. By examining session logs from some of our experimental runs on the "BBNB" machine, we were able to evaluate another easily obtainable system measure, Group Load. Group Load is a (one minute) average queue length of the number of processes belonging to a single "pie-slice" group (e.g. NSW) modified by the portion of the CPU bought by that group. Group Load is intended as a measure of the competition for that percentage of the processor allocated to serving the group, extrapolated to a whole machine. This measure is only computed on the BBN computer center machines, and was not available on the ISI configuration. In our limited tests, we found that Group Load by itself was no better at adequately "explaining" the observed delay. A summary of some of the real time response data for both the ISIC and BBNB configurations can be found in tables 1 and 2. We are actively pursuing the investigation into other measurable factors (e.g. paging characteristics) which contribute to the NSW performance behavior, as well as looking into (multiple) correlations among these factors.

Another somewhat surprising result of the log file analysis was the higher than expected variation in the measured CPU usage

for executing seemingly similar code segments. This reinforces similarly observable events from the MSG test runs (section 4.1), whereby the CPU times for successive invocations of the same event sometimes varied quite dramatically. We are now beginning a more thorough investigation of the data itself and the algorithms used in the Foreman implementation to try to explain the variation.

Table 1:

The following table reports measurement summaries for a collection of tool sessions on the ISIC "user" NSW and for test sessions on our BBNB "testing" NSW. The row labels represent number of trials (N), average elapsed real time for the operation to complete (Real Time) from the vantage point of the Foreman, standard deviation for the real time data (sd), the computed correlation of real time with load average (RT corr LD), and the computed correlation of real time with group load average (RT corr GPL). The column headings indicate the function (BEGINTOOL or ENDTOOL) and the system on which the trials were run (BBNB or ISIC). All times are recorded in seconds.

| | ISIC BEGIN | BBNB BEGIN | ISIC END | BBNB END |
|---|---|---|---|---|
| N | 213 | 64 | 118 | 61 |
| Real Time | 18.8(seconds) | 40.7(secs) | 74.5(secs) | 124.2(s) |
| sd | 25.6 | 51.2 | 75.3 | 114.9 |
| RT corr LD | 0.62 | 0.75 | 0.47 | 0.86 |
| RT corr GPL | | 0.71 | | 0.79 |

Table 2:

The following table breaks down the trial sessions on both systems into quartile bins for each operation (BEGINTOOL, ENDTOOL, OPEN and DELIVER). "MIN" is the minimum observed in each category, "I" is the 25% bound, "Median" is the 50% bound, "III" is the 75% bound, and "MAX" is the maximum observed value. All times are recorded in seconds.

| | Real Time (secs) | | LOAD AV. | | GROUP. LOAD |
|---|---|---|---|---|---|
| | ISIC | BBNB | ISIC | BBNB | BBNB |
| BEGINTOOL: | | | | | |
| MIN | 2.8 | 6.1 | 0.23 | 0.89 | 0.77 |
| I | 7.3 | 13.2 | 1.32 | 3.05 | 2.62 |
| MED | 11.1 | 24.1 | 3.22 | 4.03 | 4.27 |
| III | 18.9 | 48.7 | 5.61 | 5.35 | 6.37 |
| MAX | 271.7 | 291.7 | 18.41 | 30.70 | 26.00 |
| ENDTOOL: | | | | | |
| MIN | 11.2 | 18.5 | 0.31 | 0.35 | 0.08 |
| I | 25.6 | 47.6 | 1.86 | 3.05 | 2.32 |
| MED | 47.2 | 85.9 | 3.96 | 4.42 | 4.86 |
| III | 96.3 | 139.9 | 6.68 | 7.31 | 8.48 |
| MAX | 455.9 | 618.0 | 24.09 | 38.20 | 28.05 |
| OPEN: | | | | | |
| MIN | 3.1 | 7.9 | 0.45 | 0.76 | 0.57 |
| I | 19.1 | 29.9 | 2.14 | 2.50 | 1.74 |
| MED | 36.6 | 57.9 | 3.98 | 4.37 | 4.96 |
| III | 76.1 | 111.0 | 6.58 | 9.13 | 7.14 |
| MAX | 521.8 | 259.3 | 17.96 | 21.11 | 13.31 |
| DELIVER: | | | | | |
| MIN | 8.4 | 24.6 | 0.09 | 1.42 | 1.36 |
| I | 22.6 | 38.9 | 2.28 | 2.67 | 2.52 |
| MED | 45.9 | 92.3 | 3.93 | 8.82 | 8.04 |
| III | 75.7 | 180.3 | 7.21 | 11.96 | 10.25 |
| MAX | 507.3 | 296.8 | 25.02 | 17.99 | 10.41 |